데이터마이닝 - 의료 데이터 분석

마취중 저혈압 발생하는 환자를 사전에 찾아내기

제출일	2019-12-03	전공	컴퓨터공학과
과목	데이터마이닝	학번	20184060
담당교수	우지영 교수님	이름	안성현

< 목 차 >

과제 조사

1-1] '혈압'에 관해서 조사하기

'혈압'은 혈관을 따라 흐르는 혈액이 혈관의 벽에 주는 압력이다.

'혈압'은 '수축기 혈압'과 '이완기 혈압' 이 존재한다.

'수축기 혈압'은 온몸에 피를 보내기 위해서 심장이 대동맥에게 피를 보낼 때 발생하는 압력이다.

'수축기 혈압'이 140 이상으로 올라가면 '고혈압'이라고 한다.

이는 혈관이 안 좋아서 혈액이 찐득찐득하면 속도가 느리니, 심장이 어쩔 수 없이 대동맥에 피를 더 보내고 혈관이 받는 압력이 높아지는 경우를 말한다. (심장 근육이 안 좋은 경우, 더 많은 피를 보내기 위해 맥박이 높아지는 경우가 있다.)

고혈압이 오면 혈관은 약해지고 상처를 입을 뿐 아니라, 심장은 힘을 더 쓰니 심부전으로 갈 수도 있다.

'이완기 혈압'은 심장에 피가 채워질 때 동맥에 발생하는 압력이다.

심장에 피가 채워질 때 심장을 먹여 살리는 '관상동맥'에도 피가 채워진다. '관상동맥'에 충분하게 혈액이 공급돼야지 심장에게 산소와 영양소를 잘 공급한다.

'이완기 혈압'이 60미만이면 '저혈압'이라고 한다.

'이완기 혈압'이 지나치게 떨어지면 심장에 공급하는 피가 부족해진다. '수축기 혈압'이 올라가는 만큼 심장은 많은 피와 산소가 필요하다. 그런데 피가 채워지지 않으니 관상동맥에도 혈액 공급이 안되고 결국 심장의 산소가 부족해진다. 이또한 심부전과 심근경색으로 이어질 수 있다.

보통 '이완기 혈압' 수치보다는 '수축기 혈압' 수치가 90미만일 때 '저혈압'이라고 한다. 이 때문에 피가 많이 안돌면 뇌, 심장, 신장 등 중요 장기로 혈액을 더 공급하고 덜 중요한 장기는 혈액을 덜 공급한다. 이 과정에서 두통, 피로감, 호흡곤란이 생길 수도 있다.

결국 고혈압과 저혈압은 (수축기 혈압, 이완기 혈압)수치로 평가되고, 산소 포화도(적혈구에 의해 운반되는 산소의 양)나 맥박(심장의 박동) 등 여러 가지 상태에 영향을 미친다.

필자는 이러한 조사 과정을 통해 **(수축기 혈압, 이완기 혈압, 심박 수, 산소 포화도)를 변수로 선정**했다.

전처리 과정

2-1] 데이터 세트 R-studio로 가져오기

① 코드

```
# 디렉토리에 위치한 모든 파일 이름 가져오기
dir<-("D:/대학교 과제/SCH 2-2/데이터마이닝/마이닝 과제/homework")
file_list<-list.files(dir)
answer<-c() # 정답(정상,저혈압)을 담을 벡터
for(file in file_list){
  print(file)
  # 데이터 불러오기
  temp<-read.csv(paste(dir,file,sep="/"),header=TRUE,sep=",",
                 stringsAsFactors = FALSE)
  # 클래스(정답)은 클래스 변수에 넣기
  class<-temp$class[1]
  answer<-rbind(answer,class)}
```

② 결과물

```
[1] "homeworkreemr13_181030_080632.txt.csv"
[1] "homeworkreemr13_181030_093140.txt.csv"
[1] "homeworkreemr13_181030_113843.txt.csv"
[1] "homeworkreemr13_181030_125217.txt.csv"
[1] "homeworkreemr13_181030_135752.txt.csv"
[1] "homeworkreemr13_181120_080304.txt.csv"
[1] "homeworkreemr13_181120_100143.txt.csv"
[1] "homeworkreemr13_181127_081540.txt.csv"
[1] "homeworkreemr13_181127_094331.txt.csv"
[1] "homeworkreemr13_181127_110919.txt.csv"
[1] "homeworkreemr13_181127_123002.txt.csv"
[1] "homeworkreemr13_181127_132913.txt.csv"
[1] "homeworkreemr13_181127_143221.txt.csv"
[1] "homeworkreemr13_190102_081106.txt.csv"
[1] "homeworkreemr13_190108_075609.txt.csv"
[1] "homeworkreemr13_190108_092651.txt.csv"
[1] "homeworkreemr13_190108_113451.txt.csv"
[1] "homeworkreemr13_190115_075652.txt.csv"
[1] "homeworkreemr13_190115_092809.txt.csv"
[1] "homeworkreemr13_190115_103749.txt.csv"
[1] "homeworkreemr13_190115_114555.txt.csv"
```

▶ for문을 통해 homework로 받은 모든 csv파일을 R Studio로 불러왔다. 각 파일은 한 환자에 대한 의료 데이터 파일로, class열에는 저혈압이 발생했는가의 여부가 기재돼있다. 이는 머신러닝을 통해 구할 답이기 때문에 지도학습용 벡터로 저장할 필요가 있다. 이는 answer에 저장된다.

2-2] 파일에서 조건을 만족하는 데이터 추출하기 ①

① 코드

```
# "마춰시작일시"를 date변수에 저장하기
  m_date<-""
  for(i in 1:length(temp$item)){
    if(temp$item[i]=="마춰시작일시"){
      m_date<-temp$date1[i]
      break;
    }
  }

  # "마춰시작일시"보다 작거나 같은 데이터 가져오기
  data<-filter(temp,temp$date1!="") # 일정이 없는 데이터는 사용하지 않음

  # 가져온 데이터를 담을 벡터
  group<-c()
  item<-c()
  value<-c()
  date<-c()

  # group,item,value,date만 가져옴
  j<-1
  for(i in 1:length(data$item)){
    if(as.character(data$date1[i])<=as.character(m_date)){
      group[j]<-as.character(data$group[i])
      item[j]<-as.character(data$item[i])
      value[j]<-as.character(data$value1[i])
      date[j]<-as.character(data$date1[i])
      j=j+ 1
    }
```

```
}
set<-data.frame(group,item,value,date) # 벡터들을 데이터 프레임으로 만듦
```

② 결과물

	group	item	value	date
1	마취일반정보	Time Out Check 일시		2019-07-23 12:40
2	마취일반정보	마취시작일시		2019-07-23 12:53
3	장비사용 정보	장비사용 정보	Patient Monoter	2019-07-23 12:53
4	장비사용 정보	장비사용 정보	마취기계	2019-07-23 12:53
5	장비사용 정보	장비사용 정보	HEATER/COOLER	2019-07-23 12:53
6	장비사용 정보	장비사용 정보	BIS	2019-07-23 12:53
7	장비사용 정보	장비사용 정보	PC화면	2019-07-23 12:53
8	마취기록	마취방법>>General Anesthesia>>Circle with ET tub...	semiclosed circle	2019-07-23 12:53
9	마취기록	마취방법>>General Anesthesia>>Circle with ET tub...	PVC	2019-07-23 12:53
10	마취기록	마취방법>>General Anesthesia>>Circle with ET tube	Cuff; 4 cc	2019-07-23 12:53
11	마취기록	마취방법>>General Anesthesia>>Circle with ET tub...	22 cm at teeth	2019-07-23 12:53
12	마취기록	마취방법>>General Anesthesia>>Circle with ET tub...	2	2019-07-23 12:53
13	마취기록	마취방법>>General Anesthesia>>Circle with ET tub...	curved direct laryngoscope	2019-07-23 12:53
14	마취기록	IV access & arterial cannulation>>Peripheral v.	Rt. arm	2019-07-23 12:53
15	마취기록	Position	supine	2019-07-23 12:53
16	Progress	1	Monitor On	2019-07-23 12:4
17	Inhalational Agent	O2 [(45L/15분)]	6	2019-07-23 12:50
18	V/S	NBP-S(mmHg)	171	2019-07-23 12:45

▶ 마취 전 데이터로 마취 후 저혈압 발생 여부를 알아야 되기 때문에 '마취 시작 일시'를 먼저 찾아 보았다. 일정이 이 시점보다 작거나 같은 데이터만 추출하려고 한다.

temp변수는 R로 불러온 하나의 csv파일이다. 이 파일의 item열 중 '마취 시작 일시' 문자열이 위치한 행을 우선 찾고, 해당 행 date1열에 위치한 값을 가져오면 된다. 이 값(마취 시작 일시)을 m_date라는 변수에 담았다.

m_date보다 일정이 작거나 같은 데이터를 추출하기 전에 filter함수를 통해 일정이 없는 데이터는 제외했다.

또한 개발에 필요한 데이터는 (group,item,value1,date1)열이므로, group,item,value,date벡터만 미리 만들어 놓았다.

이후 다시 for문을 통해 파일의 내용을 순회한다. 순회 중 요소의 일정이 m_date보다 작거나 같으면 해당 행의 (group,item,value1,date1) 데이터를 앞서 선언한 벡터에 저장하였다.

모든 순회를 마칠 때까지 위 과정을 마치면 조건을 만족하는 벡터들이 완성되었다.

위 이미지는 벡터들을 합쳐서 데이터 프레임(set)으로 구축한 모습이다.

2-2] 파일에서 조건을 만족하는 데이터 추출하기 ②

① 코드

```
# item중 수축기 혈압, 이완기 혈압, 심박 수,산소 포화도만 가져옴
  NBP_S<-c()
  NBP_D<-c()
  HR<-c()
  SPO2<-c()
  a<-b<-c<-d<-1
  for(i in 1:nrow(set)){
    si<-as.character(set$item[i])
    sv<-as.character(set$value[i])
    if(si=="NBP-S(mmHg)"){
      if(sv!="LA" && sv!="RA" && sv!="RL" && sv!="-"){
        NBP_S[a]<-sv
        a<-a+ 1;}
    }
    else if(si=="NBP-D(mmHg)"){
      if(sv!="LA" && sv!="RA" && sv!="RL"&& sv!="-"){
        NBP_D[b]<-sv
        b<-b+ 1;}
    }
    else if(si=="HR (BPM)"){
      if(sv!="LA" && sv!="RA" && sv!="RL" && sv!="-"){
        HR[c]<-sv
        c<-c+ 1;}
    }
    else if(si=="SPO2 (%)"){
      if(sv!="LA" && sv!="RA" && sv!="RL" && sv!="-"){
        SPO2[d]<-sv
        d<-d+ 1;}
    }
```

```
}


# 수축기 혈압, 이완기 혈압, 심박 수, 산소포화도 평균 구하고 저장
m_nbp_s<-mean(as.integer(NBP_S))
m_nbp_d<-mean(as.integer(NBP_D))
m_hr<-mean(as.integer(HR))
m_spo2<-mean(as.integer(SPO2))
```

② 결과물

```
> m_nbp_s
[1] 172
> m_nbp_d
[1] 72
> m_hr
[1] 71.5
> m_spo2
[1] 97.5
```

▶ 데이터 프레임(set)의 모든 데이터를 쓰는 것이 아니다. 수축기 혈압(NBP-S), 이완기 혈압(NBP-D), 맥박 수(HR), 산소 포화도(SPO2)에 해당하는 데이터만 이용할 것이다.
따라서 수축기 혈압 벡터(NBP_S), 이완기 혈압 벡터(NBP_D), 맥박 수 벡터(HR), 산소 포화도 벡터(SPO2)를 미리 만들고 추출을 하였다.
추출은 set을 for문으로 순회하며 진행된다. 먼저 set의 item열에서 (“NBP-S”, “NBP-D”,”HR”, “SPO2”) 중 하나에 해당하는 문자열을 찾으면, 해당 행의 value값을 벡터에 저장하는 방식이다. csv파일을 보면 value1에 수치만 적혀 있는 것이 아니라 (“RA”,”LA”,”RL”,”-“)와 같은 문자열도 적혀 있는데 이는 수치 계산이 불가하므로 제외하였다. 이 방법으로 순회를 마치면 기대한 모든 벡터들이 완성된다.
이 벡터들의 값은 어디까지나 한 환자에 대한 데이터이다. 머신러닝을 할 때는 여러 환자의 데이터를 이용해야 하므로, 벡터 모든 값을 이용하면 다소 복잡해질 수 있다.
따라서 필자는 수축기 혈압 벡터(NBP_S), 이완기 혈압 벡터(NBP_D), 맥박 수 벡터(HR), 산소포화도 벡터(SPO2)의 값들을 모두 **평균**지었다.
평균지은 각 값들은 m_nbp_s, m_nbp_d, m_hr, m_spo2 변수에 저장된다.

2-3] 데이터프레임 형태로 구축하기

① 코드

```
answer<-c()
nbp_s<-c()
nbp_d<-c()
spo2<-c()
hr<-c()
for(file in file_list){
  print(file)
  # 데이터 불러오기
  temp<-read.csv(paste(dir,file,sep="/"),header=TRUE,sep=",",
                 stringsAsFactors = FALSE)
  # 클래스(정답)은 클래스 변수에 넣기
  class<-temp$class[1]
  answer<-rbind(answer,class)
[2-2 소스]
  nbp_s<-rbind(nbp_s,m_nbp_s)
  nbp_d<-rbind(nbp_d, m_nbp_d)
  hr<-rbind(hr,m_hr)
  spo2<-rbind(spo2,m_spo2)
}
# 데이터 합치기
patients<-data.frame("NBP_S"=nbp_s,
                     "NBP_D"=nbp_d,
                     "HR"=hr,
                     "SPO2"=spo2,
                     "class"=answer)

# 결측값은 그냥 정상 처리(나머지 변수로 분류)
for(i in 1:nrow(patients)){
```

```
  if(is.na(patients$NBP_S[i])){patients$NBP_S[i]<-120}
  if(is.na(patients$NBP_D[i])){patients$NBP_D[i]<-80}
  if(is.na(patients$HR[i])){patients$HR[i]<-60}
  if(is.na(patients$SPO2[i])){patients$SPO2[i]<-95}
}
View(patients) # 환자들 데이터
```

② 결과물

	NBP_S	NBP_D	HR	SPO2	class
m_nbp_s	155.00000	74.00000	62.00000	96.00000	hypotension
m_nbp_s.1	145.00000	64.50000	96.00000	97.00000	normal
m_nbp_s.2	138.50000	75.66667	96.00000	97.00000	hypotension
m_nbp_s.3	98.50000	58.50000	73.50000	99.50000	hypotension
m_nbp_s.4	131.85714	74.00000	64.00000	99.00000	normal
m_nbp_s.5	139.00000	80.00000	65.00000	95.00000	hypotension
m_nbp_s.6	136.00000	73.00000	52.00000	96.00000	hypotension
m_nbp_s.7	177.60000	103.20000	101.00000	96.00000	normal
m_nbp_s.8	147.50000	78.33333	67.50000	98.50000	normal
m_nbp_s.9	155.00000	69.00000	79.00000	96.00000	normal
m_nbp_s.10	119.33333	66.66667	57.00000	97.00000	normal
m_nbp_s.11	120.00000	73.00000	78.00000	98.00000	normal
m_nbp_s.12	126.00000	76.50000	85.00000	94.50000	normal
m_nbp_s.13	119.33333	65.00000	66.00000	99.00000	hypotension
m_nbp_s.14	146.00000	87.00000	83.00000	97.00000	normal
m_nbp_s.15	120.00000	67.50000	71.50000	99.00000	hypotension
m_nbp_s.16	123.00000	69.00000	93.00000	100.00000	hypotension
m_nbp_s.17	121.66667	83.66667	69.00000	99.33333	normal
m_nbp_s.18	103.00000	57.00000	78.00000	98.00000	hypotension

▶ 앞에서는 어떻게 한 환자에 대해서 데이터 추출을 했는지 설명하였다. 이번에는 어떤 로직으로 여러 환자에 대해 데이터를 추출하고 위 이미지처럼 데이터 프레임을 구축하는지 알아 보겠다.

맨 처음에 나온 (answer, nbp_s, nbp_d, spo2, hr)벡터는 차례대로

answer : 환자들의 저혈압 유무 모음(지도학습 용 벡터),

nbp_s : 환자들의 수축기 혈압 평균 모음,

nbp_d: 환자들의 이완기 혈압 평균 모음,

spo2: 환자들의 맥박 수 평균 모음,

hr: 환자들의 산소포화량 평균 모음 이다.

[2-1]에서 나온 for문을 통해서 첫 번째 파일에서 마지막 파일까지 순회도중 각 파일마다 추출한 값을 m_nbp_s, m_nbp_d, m_hr, m_spo2 변수에 저장하였다. 이 변수들의 값

은 nbp_s<-rbind(nbp_s,m_nbp_s)와 같은 방식으로 기존의 벡터에 합쳐진다.

결국 최종 남은 nbp_s벡터는 (환자들의 수축기 혈압 평균 모음)이 된다. for문이 끝나면 (nbp_s, nbp_d, hr, spo2, answer)벡터들을 차례로 합쳐서 patients라는 환자들에 대한 데이터 프레임을 만들 수 있다.

[2-2]과정에서 ("RA","LA","RL","-")을 제외하고 데이터를 추출하였으나, 혹여나 놓친 게 있을지 몰라서 아래에 for문을 추가하였다.

해당 for문은 patients프레임에서 결측값이 존재하면 정상 데이터로 분류하는 코드이다. 저혈압 발생 여부를 (수축기 혈압, 이완기 혈압, 맥박 수, 산소포화량), 총 4개의 변수로 구하기 때문에 하나가 정상 처리돼도 나머지 변수에 의해 분류할 수 있을 것이라 추정했다.

	NBP_S	NBP_D	HR	SPO2	class
m_nbp_s.293	123.00000	72.50000	62.00000	97.00000	normal
m_nbp_s.294	186.00000	90.33333	59.50000	97.50000	normal
m_nbp_s.295	139.00000	87.00000	68.00000	98.00000	normal
m_nbp_s.296	106.28571	63.57143	58.50000	98.50000	hypotension
m_nbp_s.297	126.00000	73.50000	63.50000	96.00000	hypotension
m_nbp_s.298	126.00000	79.50000	63.50000	98.50000	normal
m_nbp_s.299	162.00000	84.00000	79.00000	97.00000	hypotension
m_nbp_s.300	142.66667	91.66667	75.50000	97.50000	normal
m_nbp_s.301	111.12500	71.12500	68.00000	98.33333	hypotension
m_nbp_s.302	101.00000	73.00000	90.00000	96.00000	hypotension
m_nbp_s.303	156.00000	86.00000	62.00000	94.00000	normal
m_nbp_s.304	102.50000	66.75000	61.00000	99.50000	hypotension
m_nbp_s.305	114.75000	55.50000	49.33333	98.33333	hypotension
m_nbp_s.306	118.00000	82.50000	48.00000	97.00000	normal
m_nbp_s.307	105.76923	70.30769	83.75000	97.75000	hypotension
m_nbp_s.308	117.00000	64.00000	64.00000	99.00000	hypotension
m_nbp_s.309	209.00000	92.00000	60.00000	99.00000	normal
m_nbp_s.310	122.50000	79.00000	65.00000	96.00000	normal
m_nbp_s.311	172.00000	72.00000	71.50000	97.50000	normal

머신러닝

3-1] 학습 모델 구축하기

① 코드

```
# 훈련, 시험 데이터 프레임 만들기
train_sample<-createDataPartition(patients$class,p=0.8,list=FALSE)

train<-patients[train_sample,]
test<-patients[-train_sample,]

# 랜덤 포레스트 모델 만들기
# 10 fold 10번 반복 -> 100번 돌아감
ctrl<-trainControl(method="repeatedcv",number=10,repeats=10)
grid_rf<-expand.grid(.mtry=c(2,4,8,16))

set.seed(300)
# train함수써서 랜덤 포레스트 돌려봄
# class를 모든 변수 이용해서 학습
# trcontrol: 사용자가 지정해줄 수 있는 부분(repeatedcv를 쓰겠다.)
# 튜닝
m_rf<-train(class~.,data=train,method="rf",
            metric="Kappa",trControl=ctrl,
            tuneGrid=grid_rf)
```

▶ createDataParcition함수를 이용해서 patients데이터의 80%를 학습용 데이터 train으로 만들었다.
이후 앙상블(여러 개의 결정 트리(Decision Tree)를 결합하여 하나의 결정 트리보다 더 좋은 성능을 내는 머신러닝 기법)중에서 제일 성능이 좋은 '랜덤 포레스트' 기법으로 모델을 구축하였다.
또한 k-fold-cross-validation(교차검증)을 사용해서, 우연히 테스트 데이터의 성능이 잘 나오도록 훈련, 테스트 데이터가 분리되는 상황을 막았다.
위 과정이 끝나면 학습 모델(m_rf)이 완성된다.

3-2] 학습 후 '정확도' 구하기

① 코드

```
# 시험 보기
test$p<-predict(m_rf,test)


# test 데이터의 원래값과 예측값을 비교하는 테이블을 구함
(tb<-table(test$class,test$p))
(accuracy<-sum(diag(tb))/sum(tb))
```

② 결과물

```
> # test 데이터의 원래값과 예측값을 비교하는 테이블을 구함
> (tb<-table(test$class,test$p))

             hypotension normal
  hypotension          30      6
  normal               13     12
> (accuracy<-sum(diag(tb))/sum(tb))
[1] 0.6885246
```

```
> # test 데이터의 원래값과 예측값을 비교하는 테이블을 구함
> (tb<-table(test$class,test$p))

             hypotension normal
  hypotension          28      8
  normal               10     15
> (accuracy<-sum(diag(tb))/sum(tb))
[1] 0.704918
```

▶ m_rf 모델 성능을 측정하기 위해 accuracy를 구해 보았다. Patients(312명의 환자 데이터)의 20% 데이터로 시험을 보므로 총 61개의 결과가 나온다.

첫 번째 실행했을 때는 약 69%의 정확도를 보였다.

같은 조건으로 기계 학습을 다시 하고 시험봤을 때는 **70%의 정확도까지** 나왔다.

부록(전체 코드)

```
# 워킹 디렉토리 설정
setwd("D:/대학교 과제/SCH 2-2/데이터마이닝/마이닝 과제")


library(dplyr)
library(caret)
library(e1071)
# 필요 패키지 적용하기

# 디렉토리에 위치한 모든 파일 이름 가져오기
dir<-("D:/대학교 과제/SCH 2-2/데이터마이닝/마이닝 과제/homework")
file_list<-list.files(dir)

# 파일 불러오기 및 조건 만족하는 데이터 불러오기
answer<-c()
nbp_s<-c()
nbp_d<-c()
spo2<-c()
hr<-c()

# 모든 파일 순회해서 처리
for(file in file_list){
  print(file)
  # 데이터 불러오기
  temp<-read.csv(paste(dir,file,sep="/"),header=TRUE,sep=",",
                 stringsAsFactors = FALSE)
  # 클래스(정답)은 클래스 변수에 넣기
  class<-temp$class[1]
  answer<-rbind(answer,class)

  # "마취시작일시"를 date변수에 저장하기
```

```
m_date<-""
for(i in 1:length(temp$item)){
  if(temp$item[i]=="마취시작일시"){
    m_date<-temp$date1[i]
    break;
  }
}

# "마취시작일시"보다 작거나 같은 데이터 가져오기
data<-filter(temp,temp$date1!="") # 일정이 없는 데이터는 사용하지 않음

# 가져온 데이터를 담을 벡터
group<-c()
item<-c()
value<-c()
date<-c()

# group,item,value,date만 가져옴
j<-1
for(i in 1:length(data$item)){
  if(as.character(data$date1[i])<=as.character(m_date)){
    group[j]<-as.character(data$group[i])
    item[j]<-as.character(data$item[i])
    value[j]<-as.character(data$value1[i])
    date[j]<-as.character(data$date1[i])
    j=j+1
  }
}
set<-data.frame(group,item,value,date) # 벡터들을 데이터 프레임으로 만듦

# item중 수축기 혈압, 이완기 혈압, 심박 수,산소 포화도만 가져옴
NBP_S<-c()
NBP_D<-c()
```

```
HR<-c()
SPO2<-c()
a<-b<-c<-d<-1
for(i in 1:nrow(set)){
  si<-as.character(set$item[i])
  sv<-as.character(set$value[i])
  if(si=="NBP-S(mmHg)"){
    if(sv!="LA" && sv!="RA" && sv!="RL" && sv!="-"){
      NBP_S[a]<-sv
      a<-a+ 1;}
  }
  else if(si=="NBP-D(mmHg)"){
    if(sv!="LA" && sv!="RA" && sv!="RL"&& sv!="-"){
      NBP_D[b]<-sv
      b<-b+ 1;}
  }
  else if(si=="HR (BPM)"){
    if(sv!="LA" && sv!="RA" && sv!="RL" && sv!="-"){
      HR[c]<-sv
      c<-c+ 1;}
  }
  else if(si=="SPO2 (%)"){
    if(sv!="LA" && sv!="RA" && sv!="RL" && sv!="-"){
      SPO2[d]<-sv
      d<-d+ 1;}
  }
}

# 수축기 혈압, 이완기 혈압, 심박 수, 산소포화도 평균 구하고 저장
m_nbp_s<-mean(as.integer(NBP_S))
m_nbp_d<-mean(as.integer(NBP_D))
m_hr<-mean(as.integer(HR))
m_spo2<-mean(as.integer(SPO2))
```

```
  nbp_s<-rbind(nbp_s,m_nbp_s)
  nbp_d<-rbind(nbp_d, m_nbp_d)
  hr<-rbind(hr,m_hr)
  spo2<-rbind(spo2,m_spo2)
}

# 데이터 합치기
patients<-data.frame("NBP_S"=nbp_s,
                     "NBP_D"=nbp_d,
                     "HR"=hr,
                     "SPO2"=spo2,
                     "class"=answer)

# 결측값은 그냥 정상 처리(나머지 변수로 분류)
for(i in 1:nrow(patients)){
  if(is.na(patients$NBP_S[i])){patients$NBP_S[i]<-120}
  if(is.na(patients$NBP_D[i])){patients$NBP_D[i]<-80}
  if(is.na(patients$HR[i])){patients$HR[i]<-60}
  if(is.na(patients$SPO2[i])){patients$SPO2[i]<-95}
}
View(patients) # 환자들 데이터

# 훈련, 시험 데이터 프레임 만들기
train_sample<-createDataPartition(patients$class,p=0.8,list=FALSE)

train<-patients[train_sample,]
test<-patients[-train_sample,]

# 랜덤 포레스트 모델 만들기
# 10 fold 10번 반복 -> 100번 돌아감
ctrl<-trainControl(method="repeatedcv",number=10,repeats=10)
grid_rf<-expand.grid(.mtry=c(2,4,8,16))
```

```
set.seed(300)
# train함수써서 랜덤 포레스트 돌려봄
# class를 모든 변수 이용해서 학습
# trcontrol: 사용자가 지정해줄 수 있는 부분(repeatedcv를 쓰겠다.)
# 튜닝
m_rf<-train(class~.,data=train,method="rf",
                metric="Kappa",trControl=ctrl,
                tuneGrid=grid_rf)

# 시험 보기
test$p<-predict(m_rf,test)

# test 데이터의 원래값과 예측값을 비교하는 테이블을 구함
(tb<-table(test$class,test$p))
(accuracy<-sum(diag(tb))/sum(tb))
```